A novel working scheme for program repair

**Generate and Validate**

**Generate – Validate – AnalyzeErr**

Analyze error using error generalization

```c
int max1 (int x){
    r = 1;
    if(x > 0){
        r = x;
    }
    assert(r > 0);
}
```

1. \(x \leftarrow 0, r \leftarrow \perp\)
2. \(x \leftarrow 0, r \leftarrow 1\)
3. \(x \leftarrow 0, r \leftarrow 1\)

**Generate using a SAT solver**

**Validate using an SMT solver**

Prune the search space differently:
- Remove all programs that fail on same input
- Remove all programs that fail on same path
- Remove some (not all) programs
- Remove programs that only might fail

Update search priority:
- Prioritize all locations along the path
- Use fault localization for the trace, and prioritize resulting locations
- Prioritize certain changes over others, based on observations in error traces

Your ideas ... ?

[x ≤ 0] [x ≤ 1]

[x%2 = 0]